CITIZEN MANAGEMENT APPLICATION

OOP MINI PROJECT - Group 14

MEMBER:
Nguyen Thi Mai Anh - 20215178
Cao Huy Dong - 20210188
Nguyen Tan Dung - 20215186

Table of Content

INTRODUCTION

Background

The rapid population growth, along with an increase in temporary residents and vacant homes, underscores the need for an effective residential management system. This project aims to address the challenges of managing a growing population in densely populated areas, providing solutions for community administrators facing these issues.

Objective

The project aims to develop an application for administrators to manage neighborhood information and fees. This tool will streamline access to resident data and enhance the efficiency of the fee collection process for individual households.

INTRODUCTION

Scope

Key features of the application include the ability to:

- View and edit residents' personal information.
- Manage community event details and notifications.
- Add, modify, and track household fees.

Designed to be simple and user-friendly, the application focuses on efficient community information management and fee collection.

Technology Used

To build a robust and efficient system, the following technologies will be employed:

- MVC Architecture
- JavaFX for the Frontend
- PostgreSQL for Database Management

OVERALL DESCRIPTION

Purpose and Problem Requirements

Population Management

- Household and individual information is recorded by the neighborhood head using a unique family register.
- Details include household number, head of household's name, address, and individual specifics (e.g., name, birth date, occupation, ID number).
- Changes such as adding new members, updating information, or recording deaths are managed systematically.
- Temporary absence and residence certificates are issued as needed.

Fee Collection Management

- Annual sanitation fees are collected (6,000 VND per person per month).
- Contributions vary based on events (e.g., supporting veterans, Children's Day).
- The accountant compiles household lists, collects fees, and records contributions.
- Summary reports of collections and participation are generated.

OVERALL DESCRIPTION

Problem-Solving Processes

Population Management

- Personal Information: Full name, ID number, date of birth, gender, ethnicity, religion, education, occupation, contact details.
- Household Information: Head of household's name, household ID, members, address, family relationships.
- Reports:

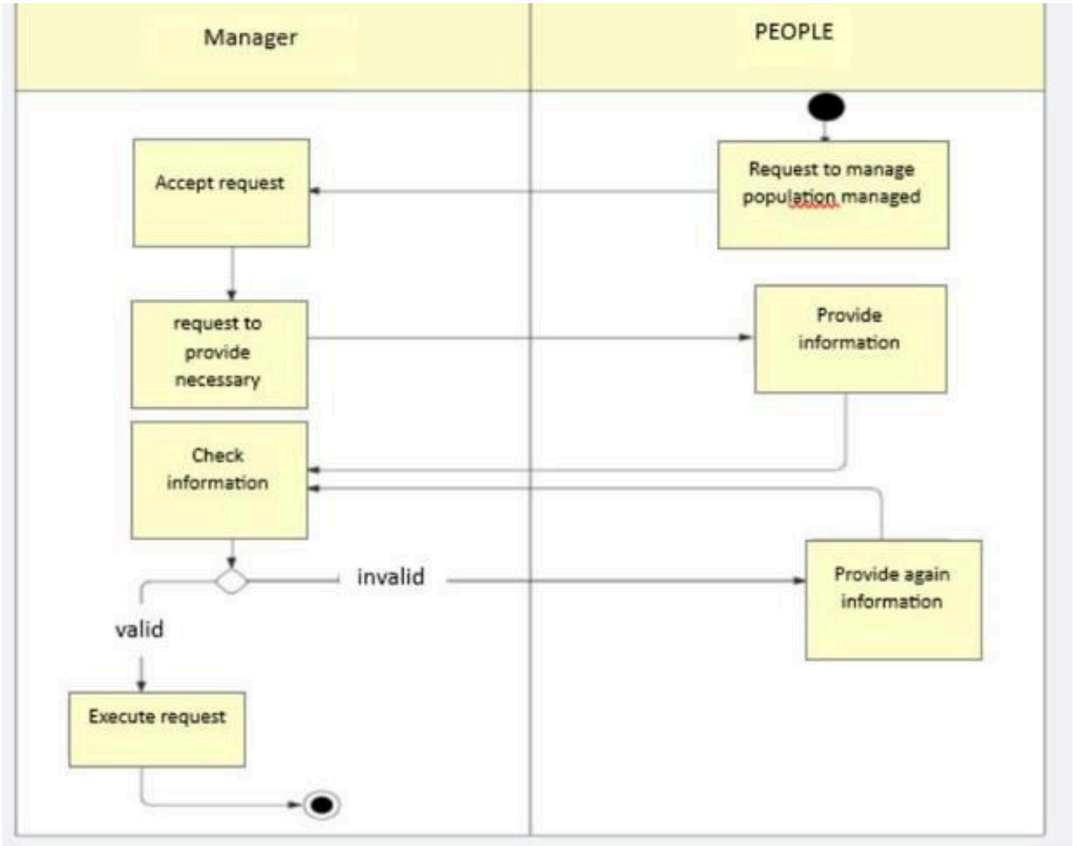
1. Temporary Residence: Report ID, individual ID, place, time, reason for moving.
2. Temporary Absence: Report ID, individual ID, place, time, reason for absence.
3. Death: Report ID, deceased's ID, reporter's ID, time of death, cause.

Fee Collection Management

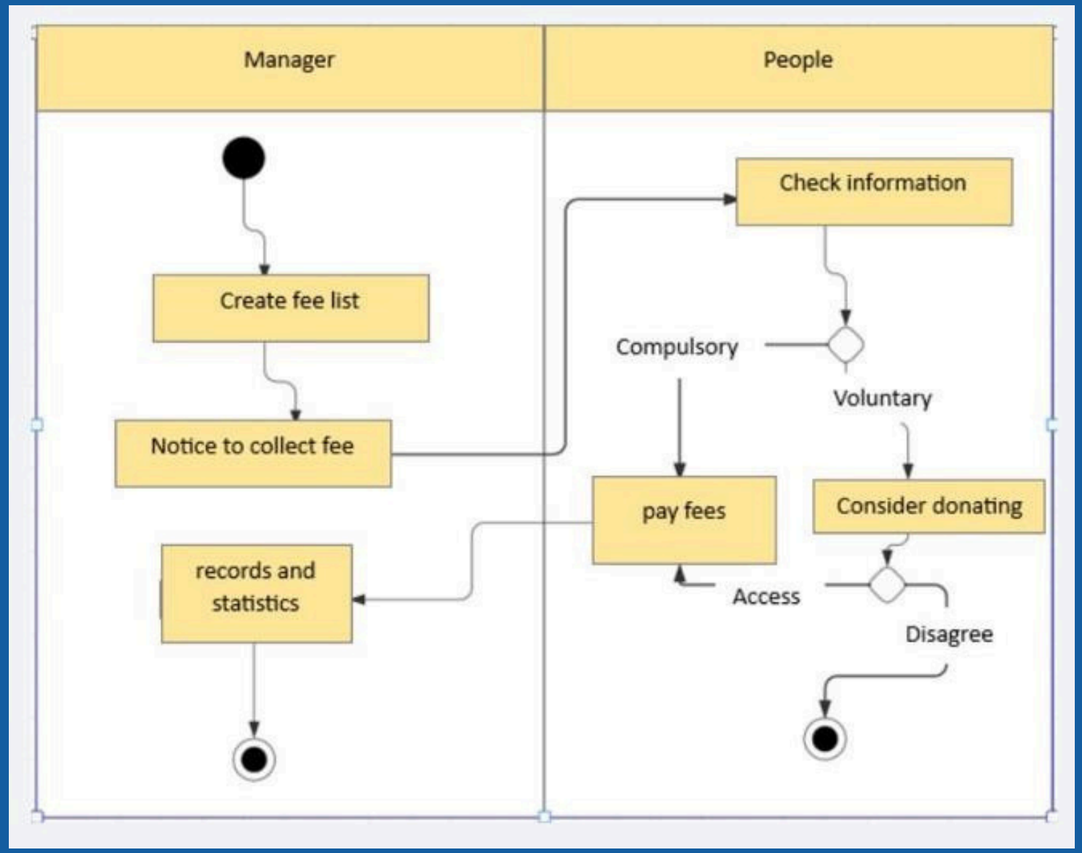
- Contributor Information: Full name, ID number, contact details (address, phone, email).
- Fee Type Information: Classification of fees (mandatory vs. contributions), purpose of contributions.
- Fee Collection Details: Specific amount, date of collection.
- Contribution and Payment History: Record of previous contributions and payments.

OVERALL DESCRIPTION

Population Management



Fee Collection Management



OVERALL DESCRIPTION

FUNCTIONAL REQUIREMENTS

Login Functionality

- Purpose: Secure administrator access.
- User Authentication: Securely store and Aauthenticate usernames/passwords.

Population Management

- Registration: Handle registrations, modifications, deletions.
- Statistics and Reporting: Generate population reports.
- Data Accuracy: Ensure accurate and timely information.
- Stored Information: Include essential personal details.

Household Management

- Actions: Add, modify, delete households.
- Statistics: Provide household data.

Fee and Contribution Management

- Categories: Create, modify, delete fee types.
- Payers: Manage payment entities.
- Information Storage: Securely store fee/contribution details.

USE CASE

FACTORS:

- User
- Population manager
- Fee collection manager

USE CASE

FACTORS:

- User
- Population manager
- Fee collection manager

DESIGN

ARCHITECTURE

Model

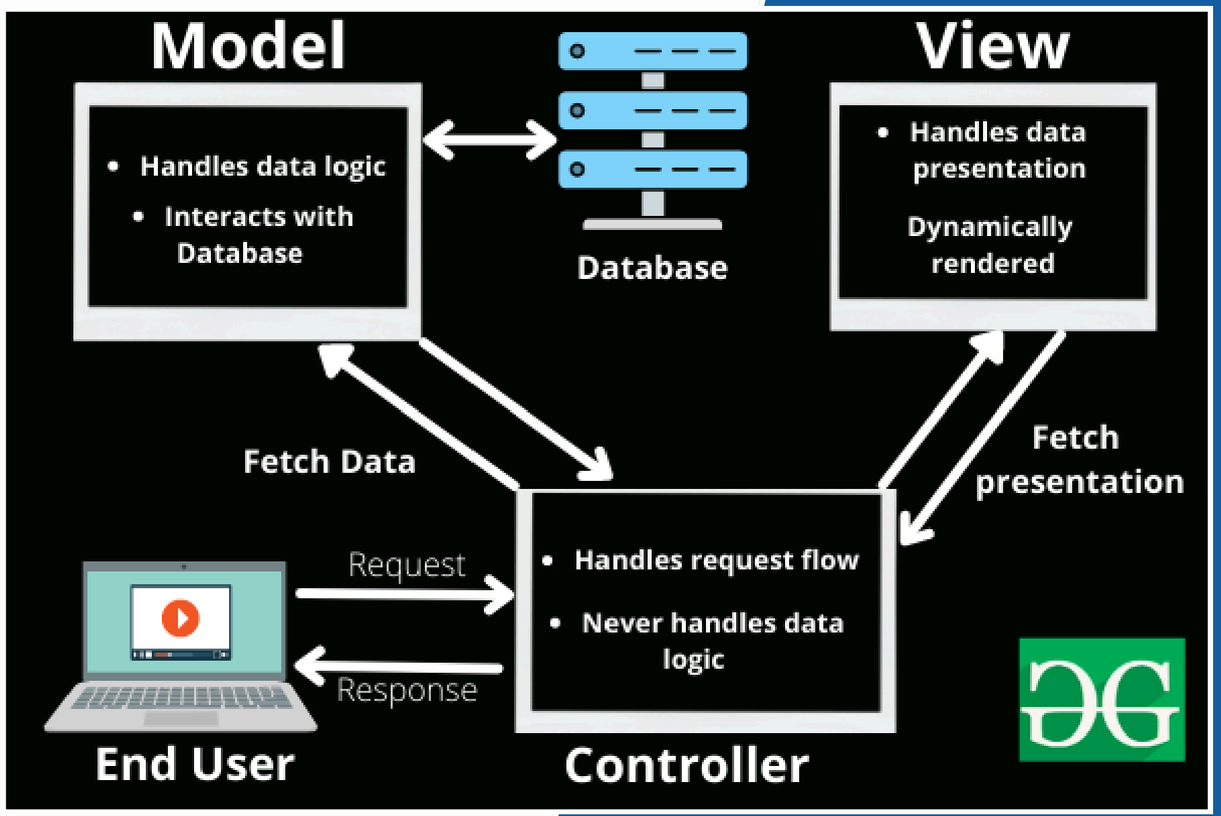
Manages data-related logic, such as retrieving and updating data from the database.

View

Handles the UI logic, including all user interface elements.

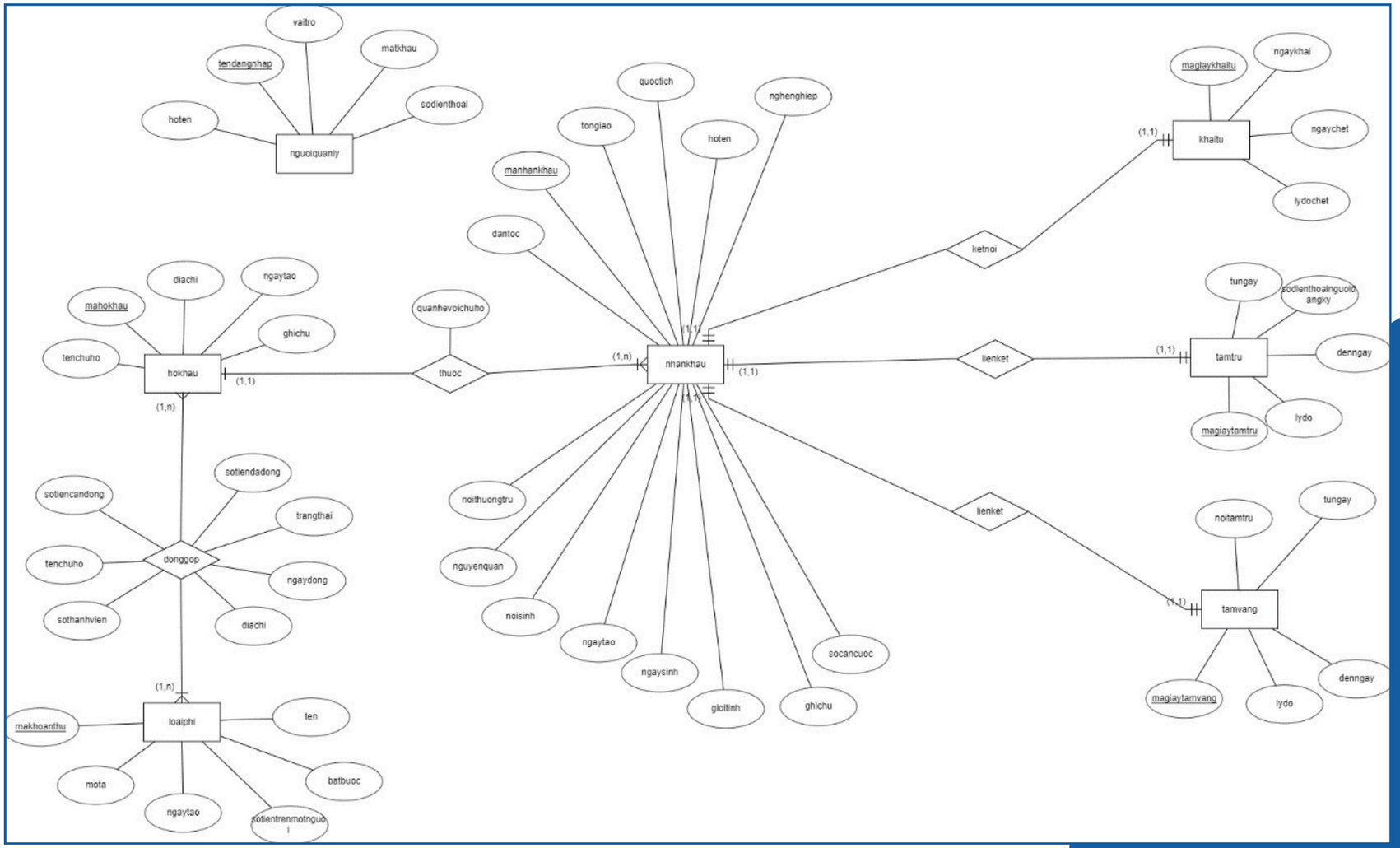
Controller

Acts as an interface between the Model and View, processing business logic, handling requests, and updating the UI.



DESIGN

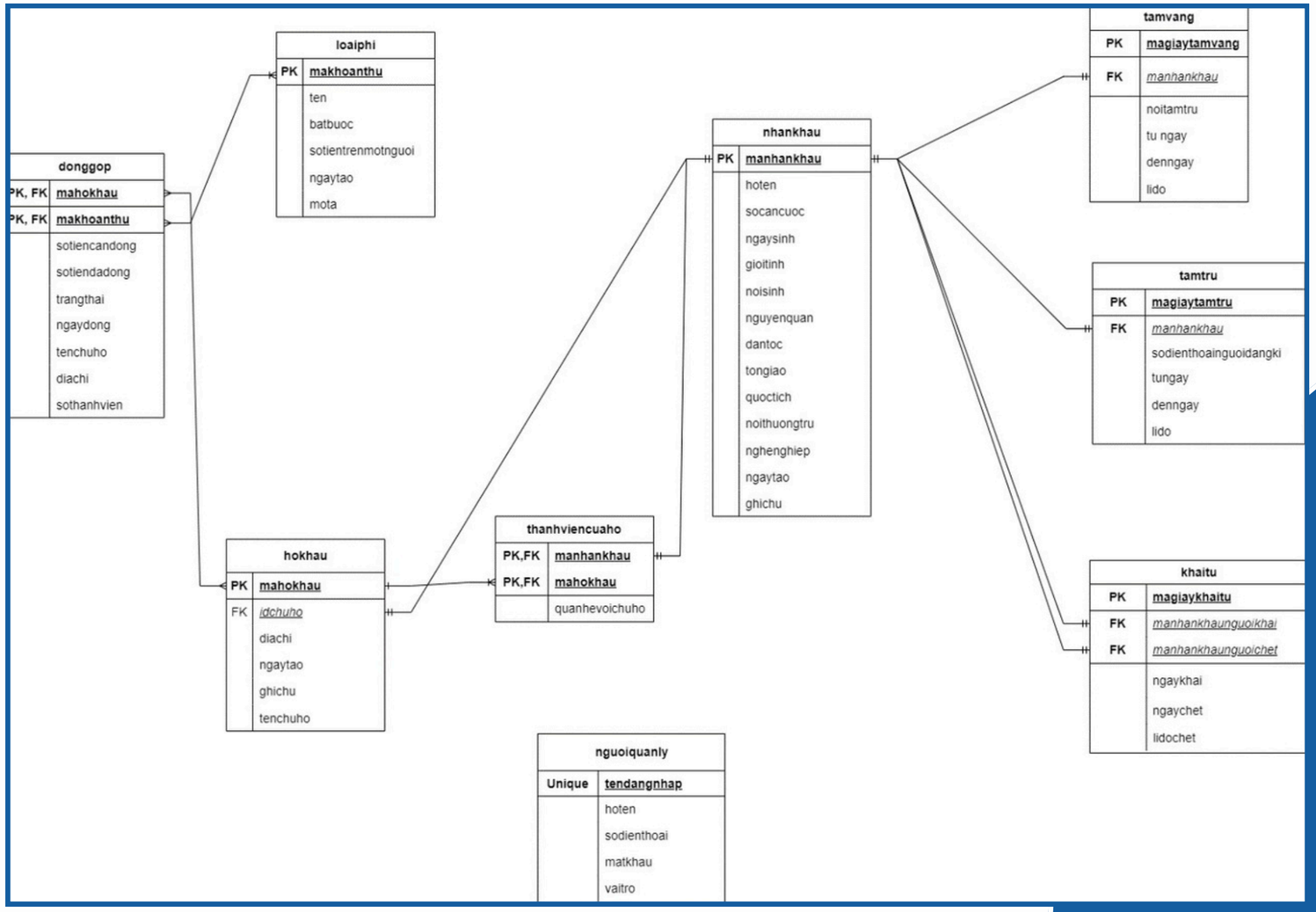
DATABASE



Entity-Relationship Diagram

DESIGN

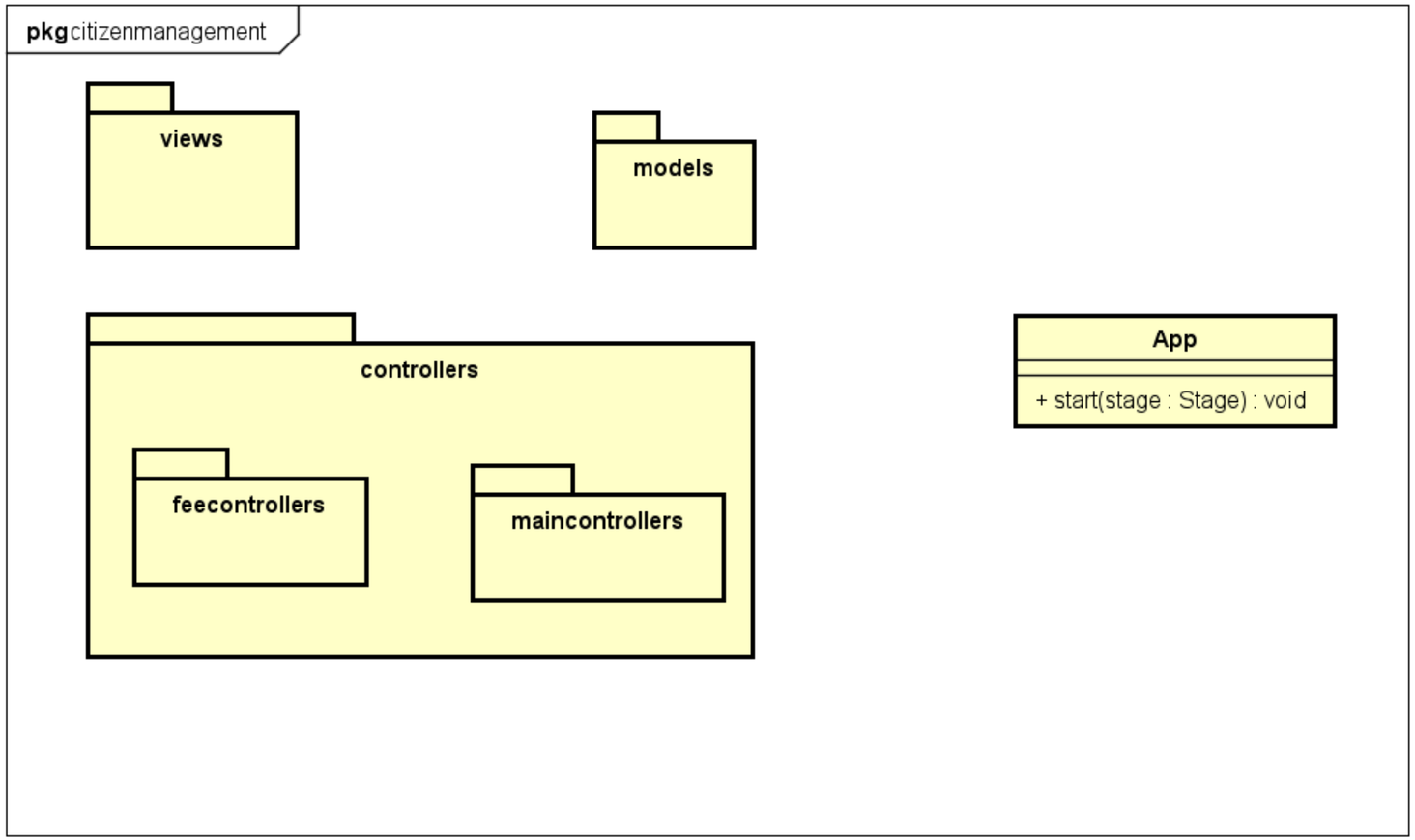
DATABASE



Class Diagram

CLASS DESIGN

General Class Diagram



CLASS DESIGN

General Class Diagram

The application begins with user requests sent as JSON to the server, then routed to appropriate Request Handlers in the Presentation Layer. JSON data is mapped to DTOs in the Business Logic Layer.

Views

GUI interfaces are built with JavaFX.

Models

- Entities retrieved from the database.
- Converted to DTOs for presentation.

Controllers

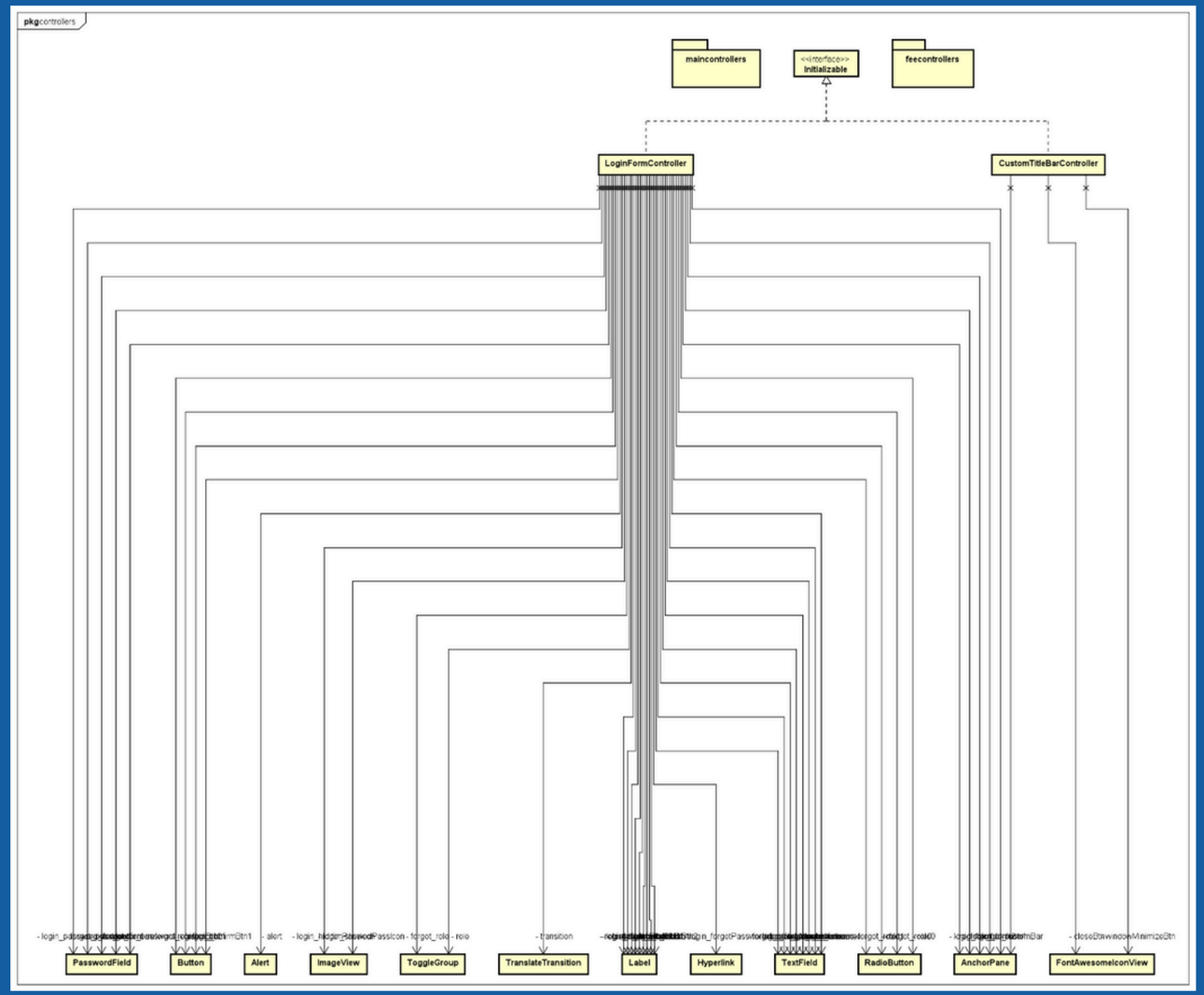
- Handle user input submission and data persistence.

Data Access Layer

- Connects to MySQL database.
- Executes SQL queries using Entity objects.
- No conversion between Entities and DTOs.

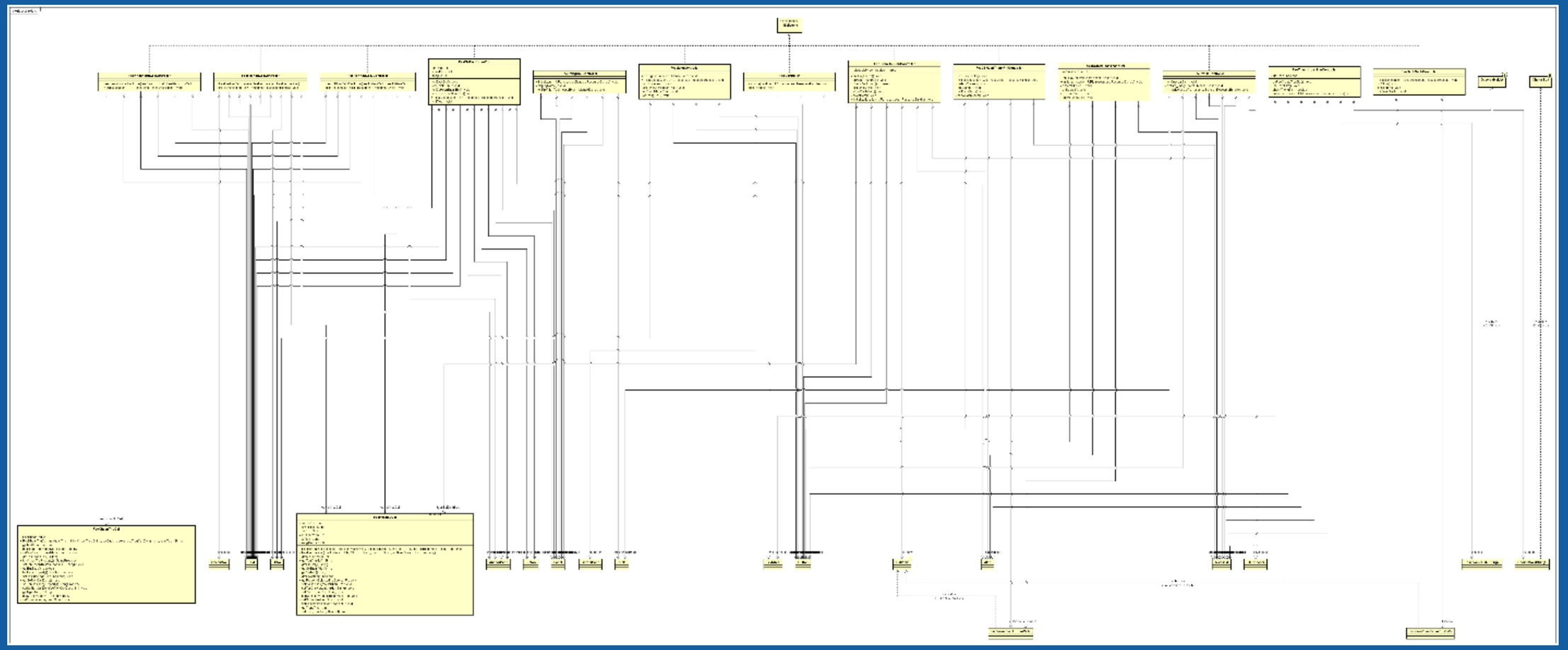
CLASS DESIGN

CONTROLLER



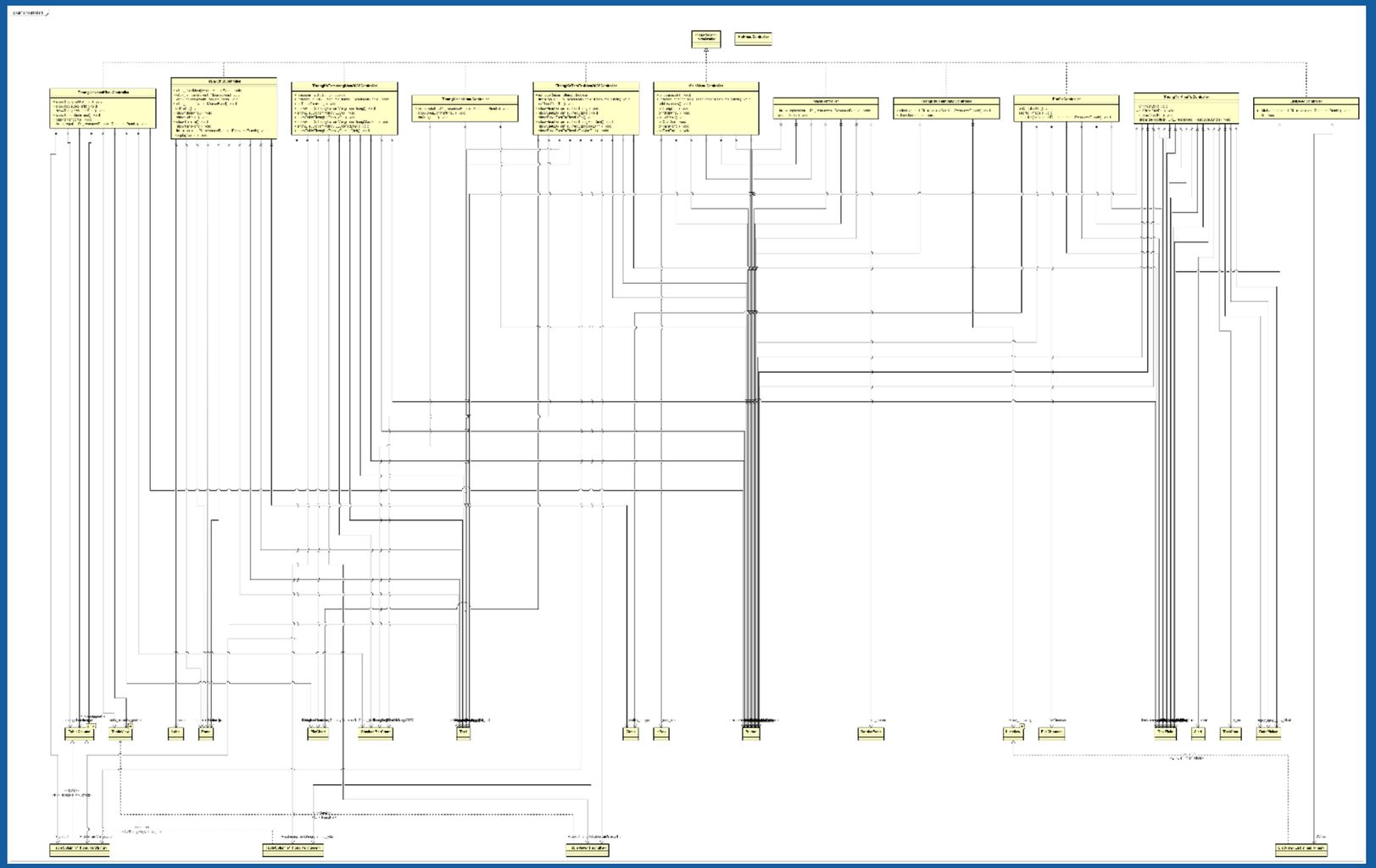
CLASS DESIGN

FEE-CONTROLLER



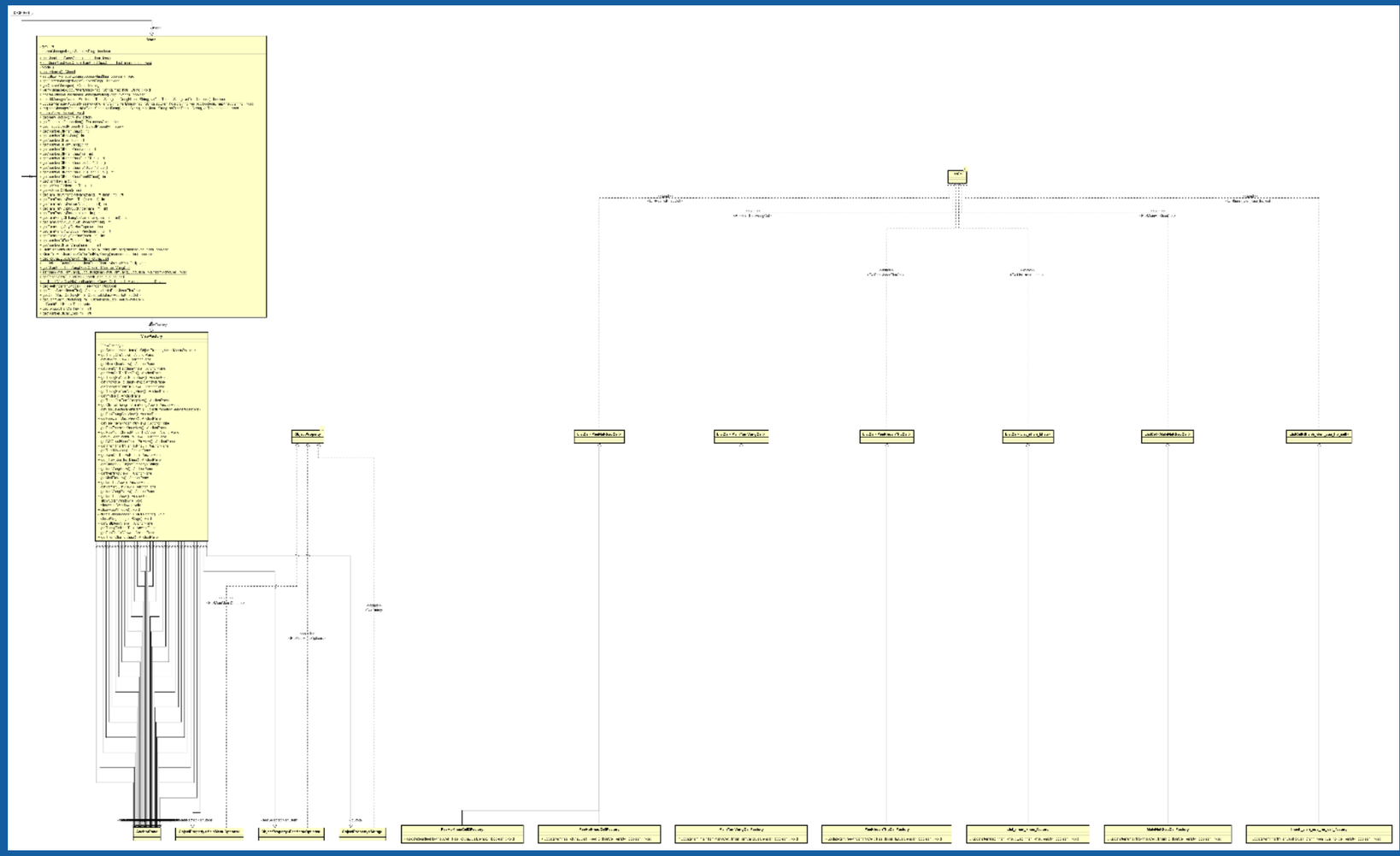
CLASS DESIGN

MAIN CONTROLLER



CLASS DESIGN

VIEW



DEMO

THANK YOU

FOR WATCHING